COP 3330: Object-Oriented Programming
Summer 2011

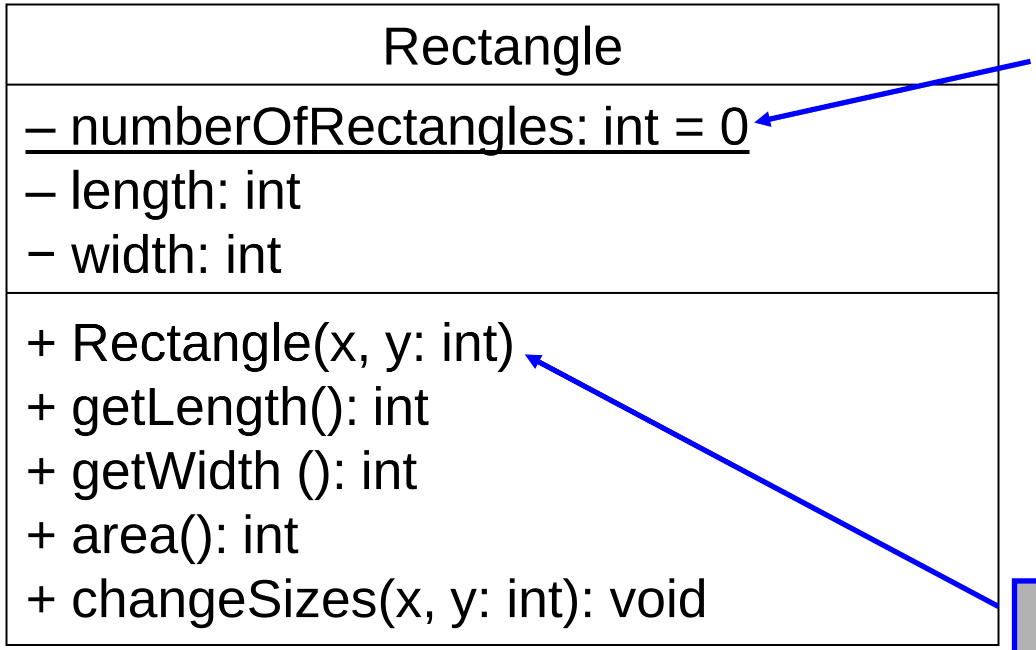
In Class Practice #1

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop3330/sum2011

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

In Class Practice #1

- Let’s convert the UML class diagram shown below into an implemented Java class and use that class to illustrate the differences between class variables/methods and instance variables/methods.

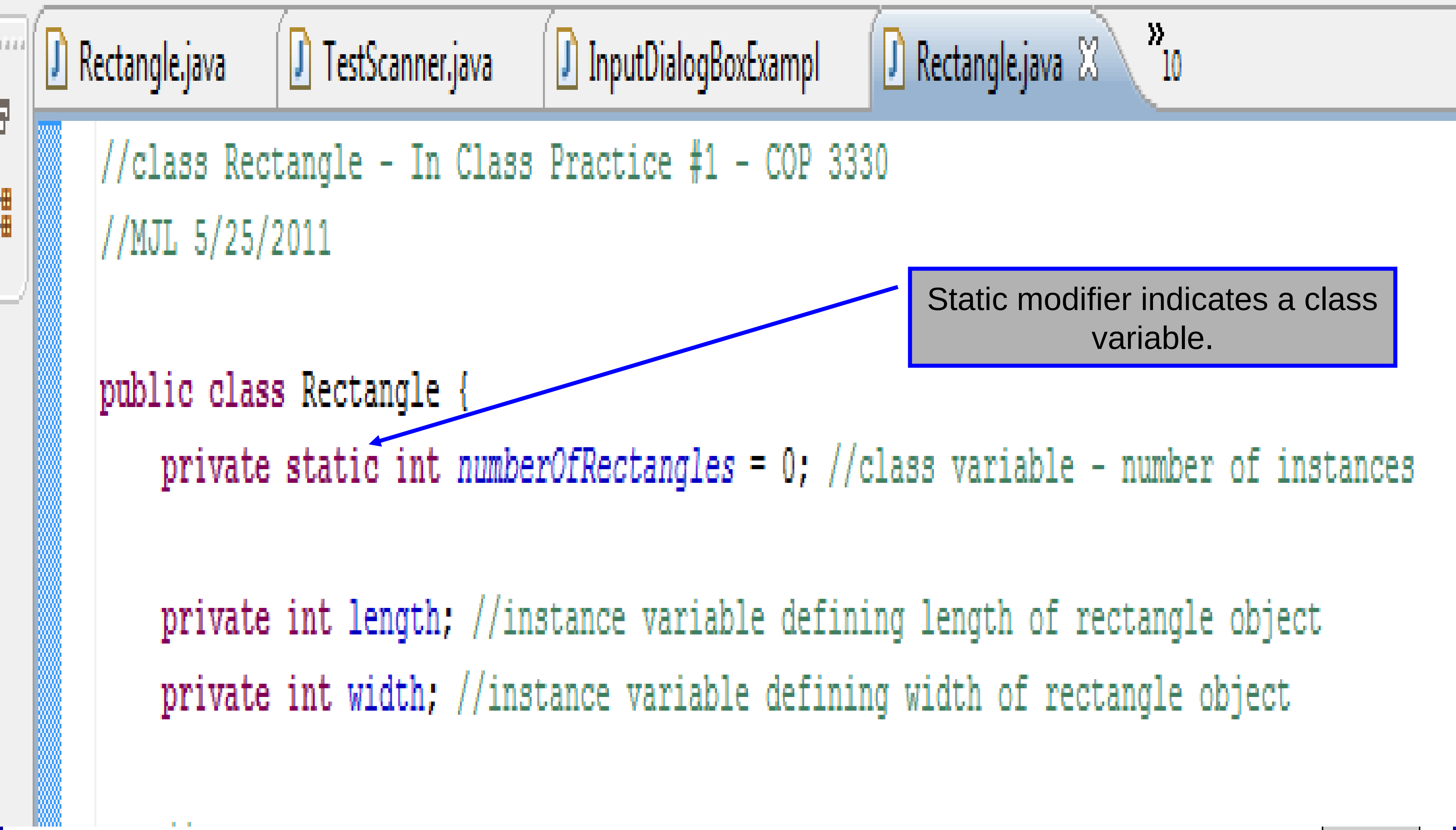


Underlining a variable or a method in a UML diagram indicates that the variable or method is a class method.

This is also the standard format for specifying a default value for a variable in UML.

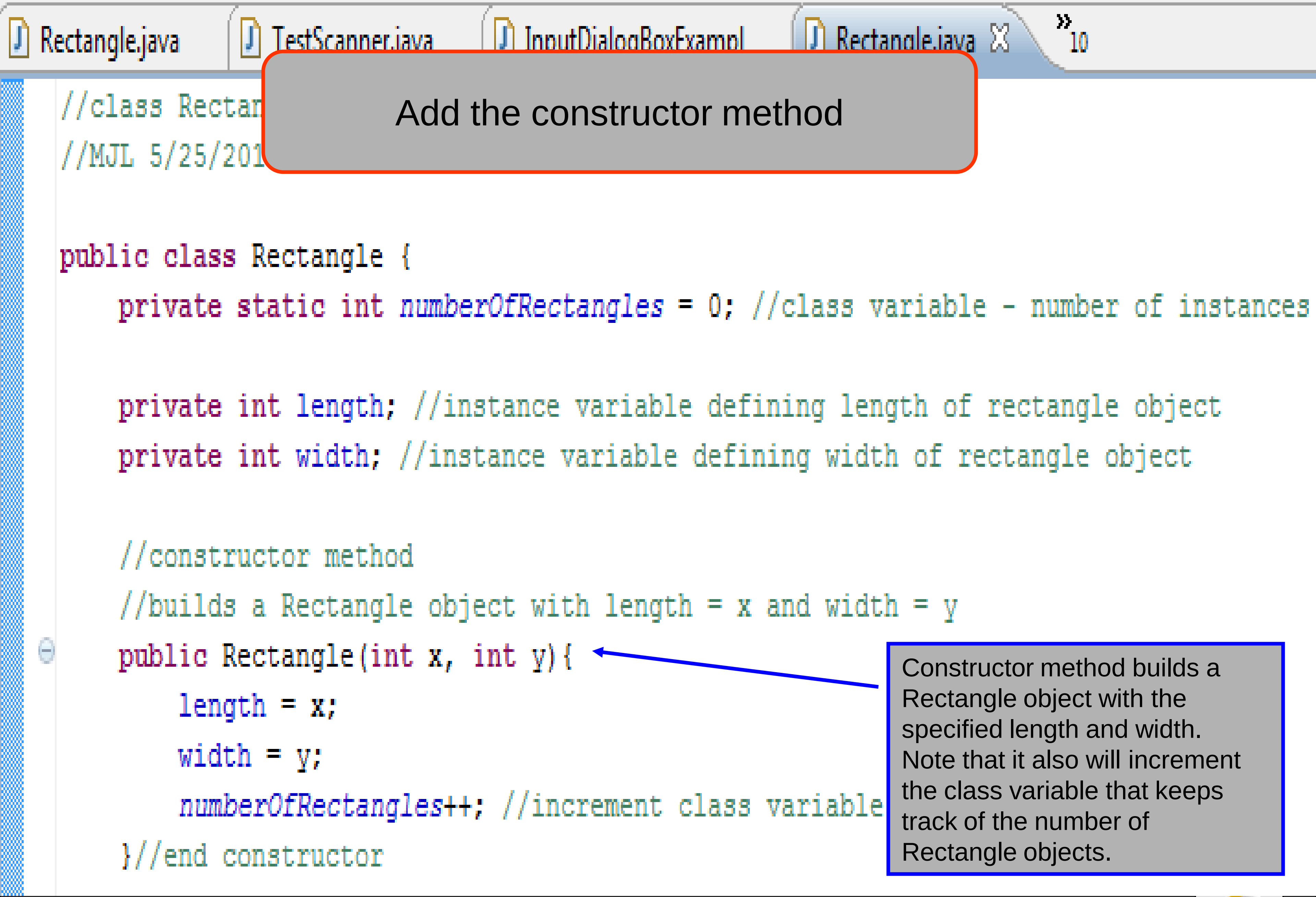
For clarity, I’ve added the constructor method, but typically this will not be included in the UML.

Create the class and add the class characteristics



```
//class Rectangle - In Class Practice #1 - COP 3330
//MJL 5/25/2011

public class Rectangle {
    private static int numberOfRectangles = 0; //class variable - number of instances

    private int length; //instance variable defining length of rectangle object
    private int width; //instance variable defining width of rectangle object
```

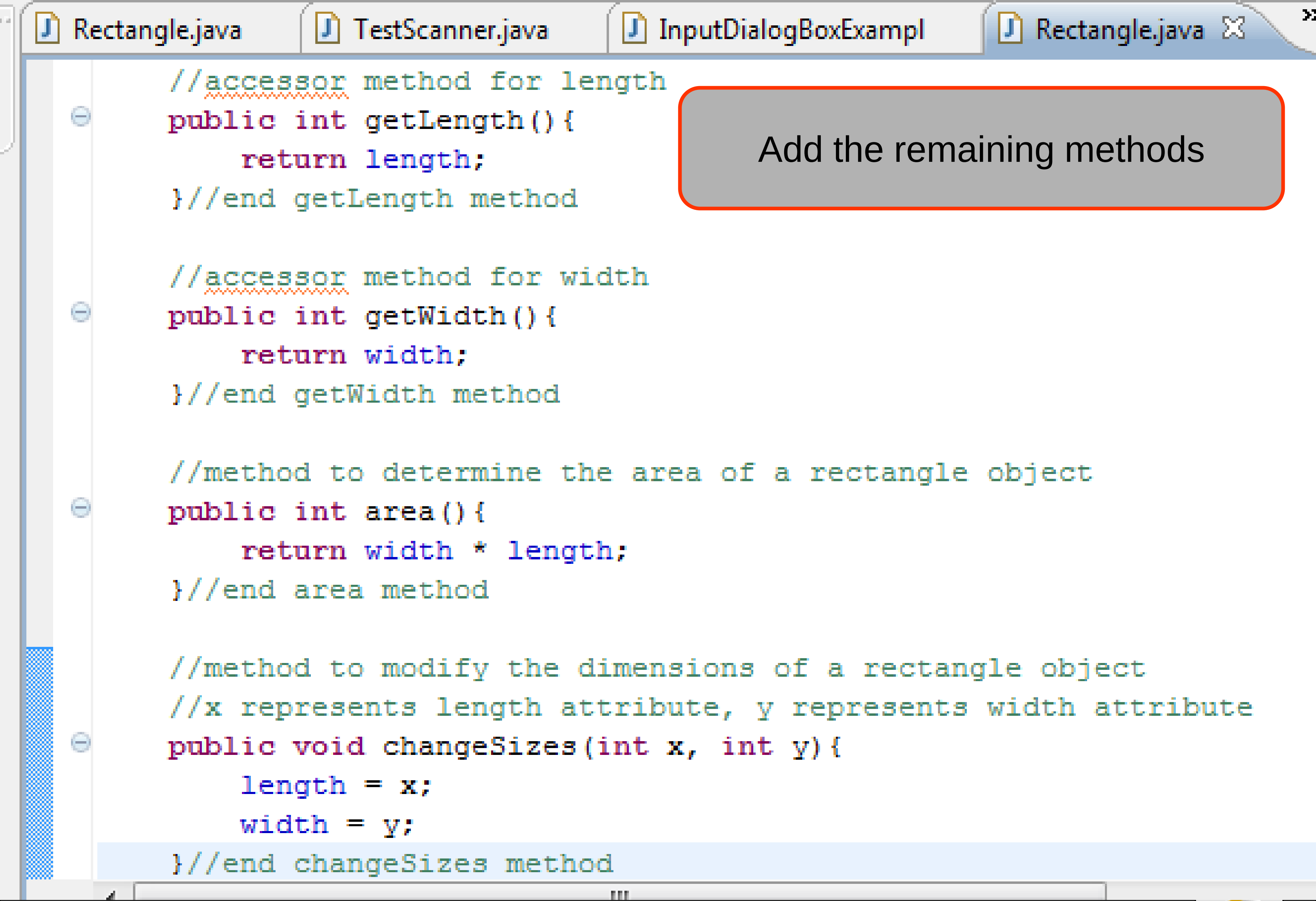
Static modifier indicates a class variable.



Add the constructor method

```
//class Rectan
//MJL 5/25/201

public class Rectangle {
    private static int numberOfRectangles = 0; //class variable - number of instances

    private int length; //instance variable defining length of rectangle object
    private int width; //instance variable defining width of rectangle object

    //constructor method
    //builds a Rectangle object with length = x and width = y
    public Rectangle(int x, int y){
        length = x;
        width = y;
        numberOfRectangles++; //increment class variable
    }//end constructor
```

Constructor method builds a Rectangle object with the specified length and width. Note that it also will increment the class variable that keeps track of the number of Rectangle objects.

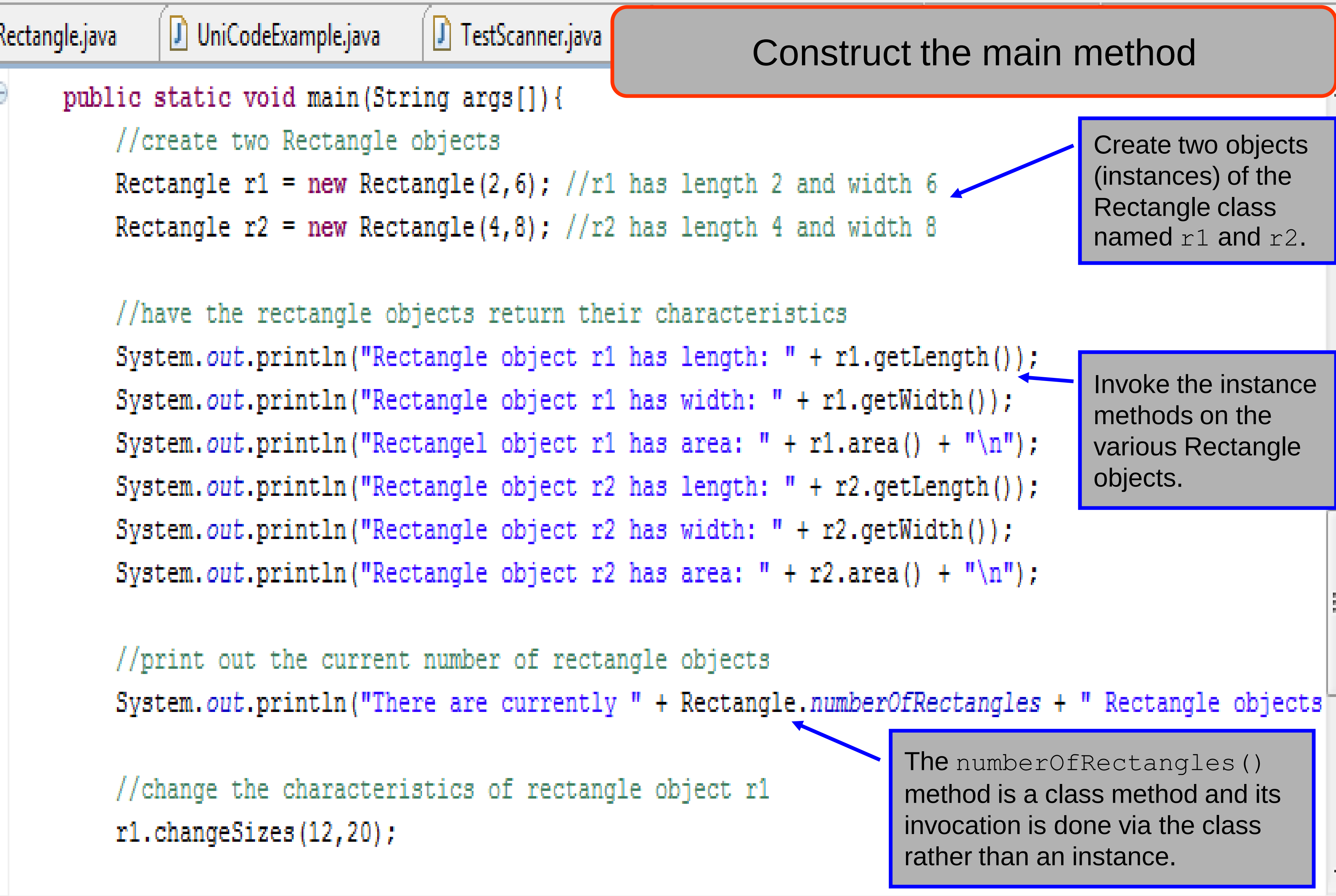


```
    //accessor method for length
    public int getLength(){
        return length;
    }//end getLength method

    //accessor method for width
    public int getWidth(){
        return width;
    }//end getWidth method

    //method to determine the area of a rectangle object
    public int area(){
        return width * length;
    }//end area method

    //method to modify the dimensions of a rectangle object
    //x represents length attribute, y represents width attribute
    public void changeSizes(int x, int y){
        length = x;
        width = y;
    }//end changeSizes method
```

Add the remaining methods

Rectangle.java | UniCodeExample.java | TestScanner.java

Construct the main method

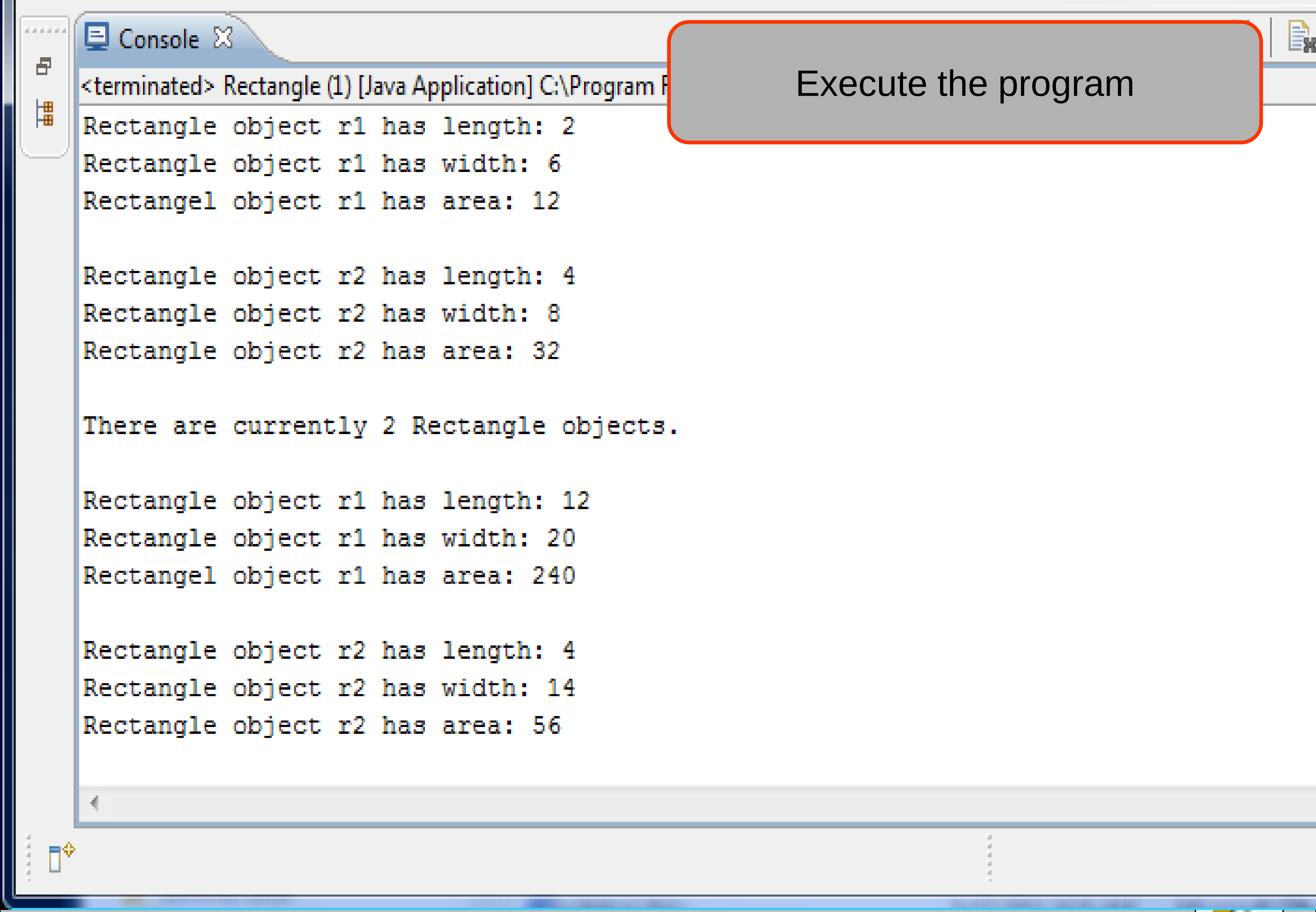


```
public static void main(String args[]){
    //create two Rectangle objects
    Rectangle r1 = new Rectangle(2,6); //r1 has length 2 and width 6
    Rectangle r2 = new Rectangle(4,8); //r2 has length 4 and width 8

    //have the rectangle objects return their characteristics
    System.out.println("Rectangle object r1 has length: " + r1.getLength());
    System.out.println("Rectangle object r1 has width: " + r1.getWidth());
    System.out.println("Rectangel object r1 has area: " + r1.area() + "\n");
    System.out.println("Rectangle object r2 has length: " + r2.getLength());
    System.out.println("Rectangle object r2 has width: " + r2.getWidth());
    System.out.println("Rectangle object r2 has area: " + r2.area() + "\n");

    //print out the current number of rectangle objects
    System.out.println("There are currently " + Rectangle.numberOfRectangles + " Rectangle objects

    //change the characteristics of rectangle object r1
    r1.changeSizes(12,20);
```

Create two objects (instances) of the Rectangle class named `r1` and `r2`.

Invoke the instance methods on the various Rectangle objects.

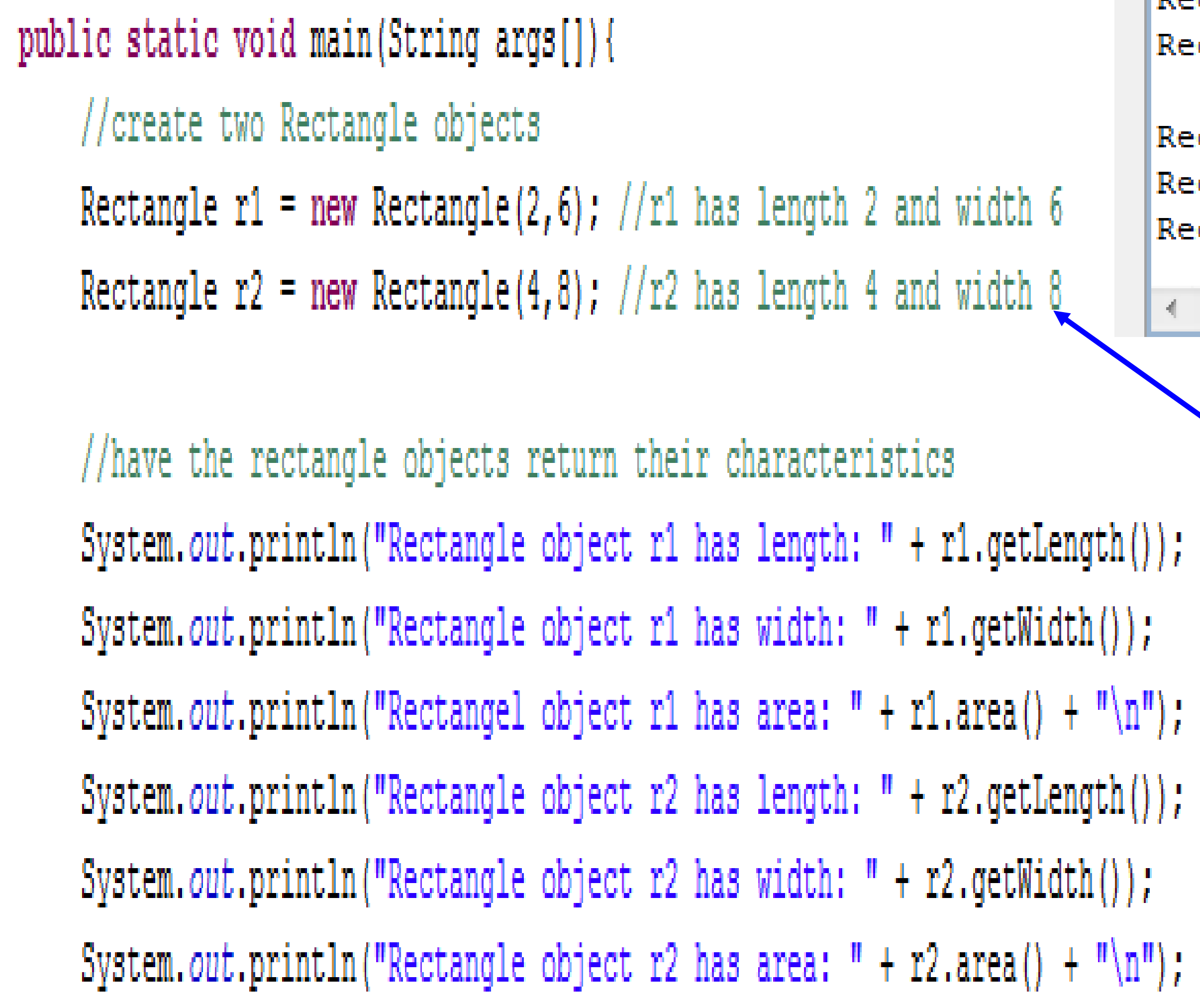
The `numberOfRectangles()` method is a class method and its invocation is done via the class rather than an instance.



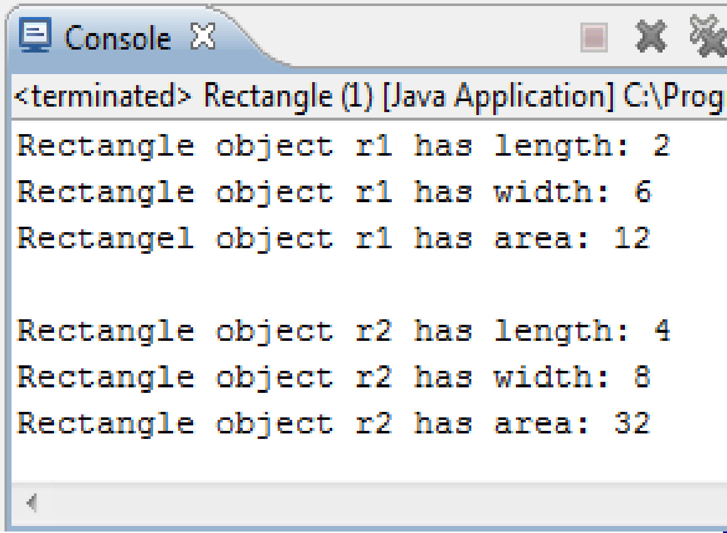
Execute the program

```
<terminated> Rectangle (1) [Java Application] C:\Program F
Rectangle object r1 has length: 2
Rectangle object r1 has width: 6
Rectangel object r1 has area: 12

Rectangle object r2 has length: 4
Rectangle object r2 has width: 8
Rectangle object r2 has area: 32

There are currently 2 Rectangle objects.

Rectangle object r1 has length: 12
Rectangle object r1 has width: 20
Rectangel object r1 has area: 240

Rectangle object r2 has length: 4
Rectangle object r2 has width: 14
Rectangle object r2 has area: 56
```

Why the output looks like it does. . .



```
public static void main(String args[]){
    //create two Rectangle objects
    Rectangle r1 = new Rectangle(2,6); //r1 has length 2 and width 6
    Rectangle r2 = new Rectangle(4,8); //r2 has length 4 and width 8

    //have the rectangle objects return their characteristics
    System.out.println("Rectangle object r1 has length: " + r1.getLength());
    System.out.println("Rectangle object r1 has width: " + r1.getWidth());
    System.out.println("Rectangel object r1 has area: " + r1.area() + "\n");
    System.out.println("Rectangle object r2 has length: " + r2.getLength());
    System.out.println("Rectangle object r2 has width: " + r2.getWidth());
    System.out.println("Rectangle object r2 has area: " + r2.area() + "\n");
```



```
<terminated> Rectangle (1) [Java Application] C:\Prog
Rectangle object r1 has length: 2
Rectangle object r1 has width: 6
Rectangel object r1 has area: 12

Rectangle object r2 has length: 4
Rectangle object r2 has width: 8
Rectangle object r2 has area: 32
```

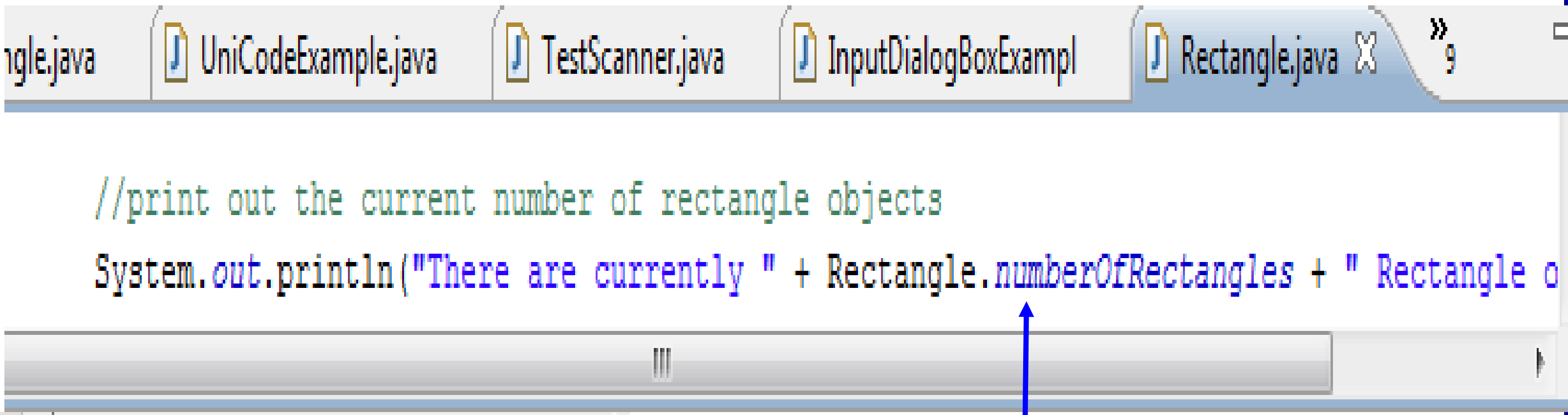
First part of the output:

The constructor method created two `Rectangle` objects named `r1` and `r2`.

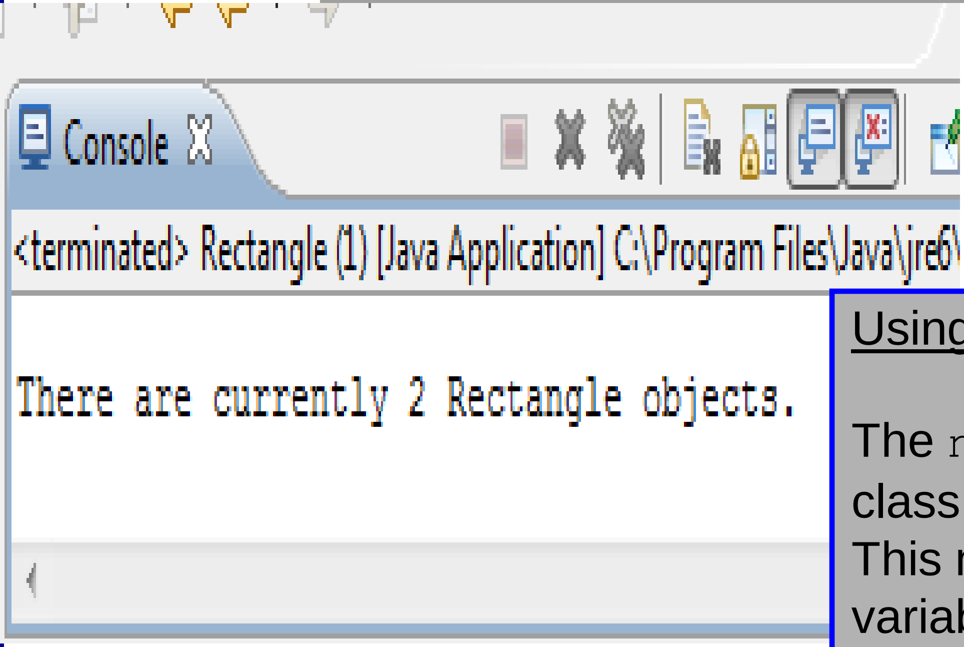
When `r1` was created the `numberOfRectangles` static variable was incremented to 1. When `r2` was created the `numberOfRectangles` static variable was incremented to 2.

Then the accessor methods `getLength()` and `getWidth()` , and the `getArea()` method were invoked on each of the objects.

Why the output looks like it does. . .



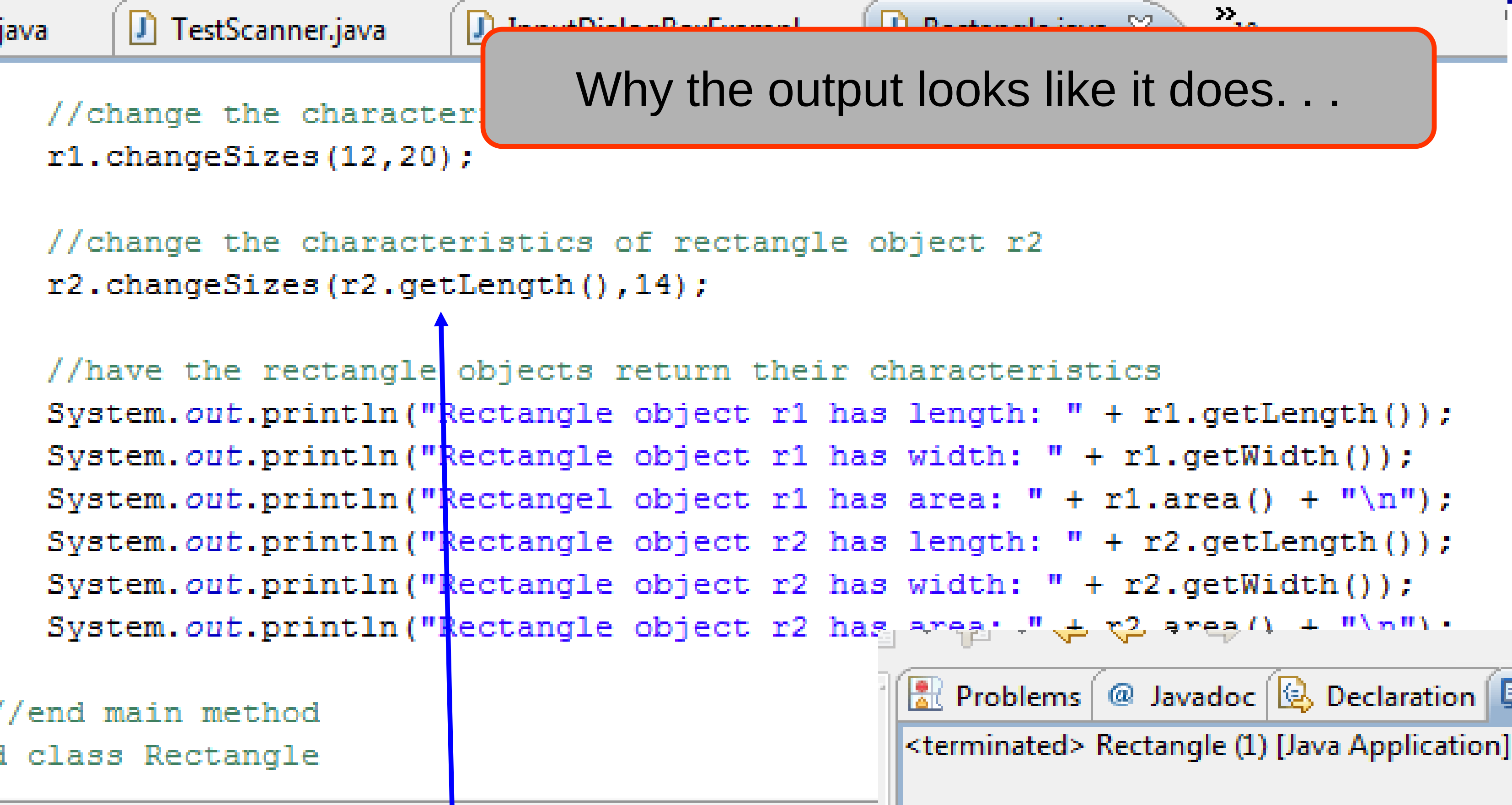
```
//print out the current number of rectangle objects
System.out.println("There are currently " + Rectangle.numberOfRectangles + " Rectangle o
```



```
<terminated> Rectangle (1) [Java Application] C:\Program Files\Java\jre6\

There are currently 2 Rectangle objects.
```

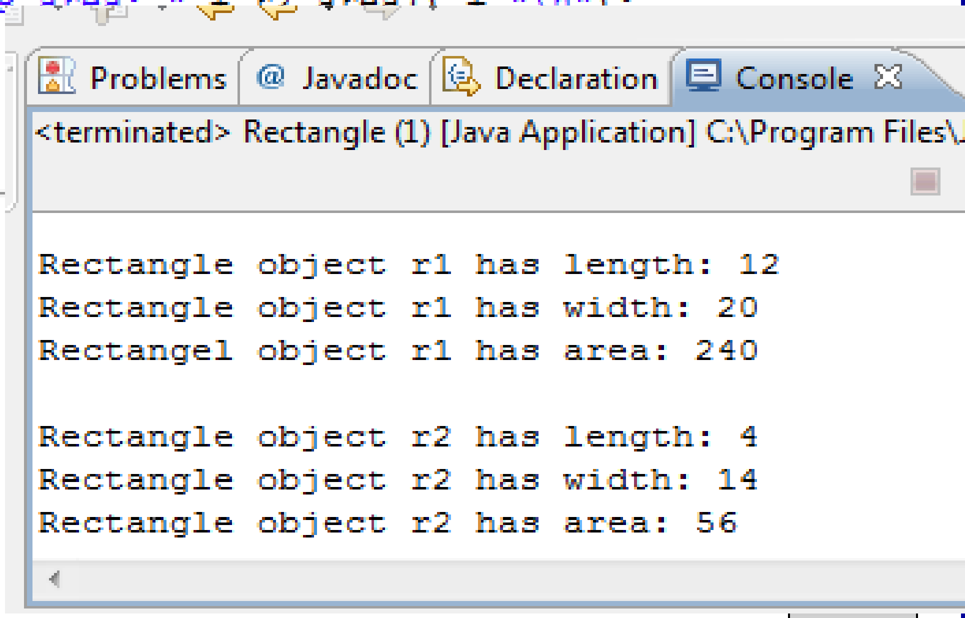
<u>Using a class variable</u>

The `numberOfRectangles` variable is a class variable (it has the static modifier). This means that access to the value of this variable must be through the class rather than through a instance of the class.



```
//change the character
r1.changeSizes(12,20);

//change the characteristics of rectangle object r2
r2.changeSizes(r2.getLength(),14);

//have the rectangle objects return their characteristics
System.out.println("Rectangle object r1 has length: " + r1.getLength());
System.out.println("Rectangle object r1 has width: " + r1.getWidth());
System.out.println("Rectangel object r1 has area: " + r1.area() + "\n");
System.out.println("Rectangle object r2 has length: " + r2.getLength());
System.out.println("Rectangle object r2 has width: " + r2.getWidth());
System.out.println("Rectangle object r2 has area: " + r2.area() + "\n");

//end main method
d class Rectangle
```

Why the output looks like it does. . .



```
<terminated> Rectangle (1) [Java Application] C:\Program Files\

Rectangle object r1 has length: 12
Rectangle object r1 has width: 20
Rectangel object r1 has area: 240

Rectangle object r2 has length: 4
Rectangle object r2 has width: 14
Rectangle object r2 has area: 56
```

Last part of the output:

Both objects `r1` and `r2` invoke their `changeSizes()` method and modify their characteristics. Object `r1` does this by supplying two new values. Object `r2` does this by invoking its own `getLength()` method to return and reuse its current length value and supplies a new width value.